VLBA Pipeline User Manual

Author: Jared Crossley
Author: Bill Cotton
Author: Gareth Hunt
Organization: National Radio Astronomy Observatory
Contact: Jared Crossley

Date: 1 March 2012

Contents

1 Introduction

The VLBA data reduction pipeline is a software pipeline that uses the Obit algorithmic development system and AIPS. The goal of the pipeline is to automate the

- download of selected raw VLBA data,

- filling of the data into the Obit/AIPS catalog,
- editing, calibration, and imaging of the data, and
- (optionally) staging of the pipeline data products for validation and subsequent ingestion into the NRAO archive.

Currently, the pipeline works only on continuum data. This manual describes the installation and operation of the pipeline software. For more details on the pipeline data reduction process see VLBA Pipeline: Outline of Data Reduction Heuristics.

A PDF version of this document is available on the Web. The source can be obtained from the `ObitDoc` svn repository. Run `make` to format this manual as PDF and HTML:

```
$ svn co https://svn.cv.nrao.edu/svn/ObitDoc/VLBAPipeline
$ cd VLBAPipeline
$ make
```

2 Download and Installation

The VLBA pipeline runs within Obit, and has been tested using Obit subversion revision 400. The pipeline may work with earlier or later revisions, but there are no guarantees. *We therefore recommend that you install revision 400 of the Obit binary*, which can be found from the Obit home page. When the Obit installation is complete and you have sourced the Obit setup script (and AIPS setup script, if necessary) you will be able to start `ObitTalk` from the command line. Now you are ready to install and run the VLBA pipeline.

To checkout a copy of the pipeline run the following subversion command:

```
$ svn co https://svn.cv.nrao.edu/svn/VLBApipeline
```

Before starting the pipeline, environment variables `PYTHONPATH`, `VLBAPIPE`, and `FITSDIR` must be set properly. Both bash and C-shell script templates have been provided for this purpose. Open one of the setup script templates, setup.sh.default or setup.csh.default, and follow the instructions in the file (The bash script is shown here, but the instructions are the same in either case.):

```
#! /bin/env bash
#
# This script initializes the environment for the VLBA Pipeline.  The script is
# distributed as a template (setup.sh.default).
#
# 1) Copy the template into a new file:
#
#      $ cp setup.sh.default setup.sh
#
```

```
# 2) Edit the new file so the environment variables declared below are set to
#    appropriate paths for your system.
#
# 3) Source the new file before starting the pipeline:
#
#      $ source setup.sh
#

# Set VLBAPIPE to the directory containing the pipeline code.
export VLBAPIPE="/export/home/cv-pipe-a/jcrossle/VLBApipeline"
# Set FITSDIR to a directory for storing archive downloads.
export FITSDIR="/lustre/aoc/users/jcrossle/fits"

# Set DESTDIR to a directory where pipeline products will be moved. Leave
# commented if you do not want products moved.
# export DESTDIR="/lustre/aoc/users/jcrossle/VLBAPipeProducts/check"

# Set umask to allow group write permissions. Leave commented if your pipeline
# products do not need to have group write permissions.
# umask u=rwx,g=rwx,o=rx # equivalent to 0002

export PYTHONPATH="$VLBAPIPE:$PYTHONPATH"
export PATH={$PATH}:{$VLBAPIPE}
```

3 Quick Start

Once the pipeline is installed, the quickest way to use the pipeline is through the continuum pipeline wrapper, `VLBAContPipeWrap.py`. The wrapper queries the archive, downloads archive data, sets up the pipeline input files and starts the pipeline script. The current version of the pipeline wrapper has the following limitations:

- The wrapper does not work on proprietary data. This requires security measures not yet in place.
- The wrapper must be run from a computer that has access to the NRAO AOC Network File System so the archive can write files directly to the `FITSDIR` directory.[1]

By default, the wrapper requires an AIPS setup script be present in the local directory. Copy the AIPSSetup.py script from the Obit `scripts` directory[2] and change the content to conform to your system and directory structure. Some of the things that you should pay attention to are:

- Setting the AIPS-data and FITS directories using variables `adirs` and `fdirs`. Several example initializations of `adirs` and `fdirs` are provided; after setting the appropriate values for your system, remove or comment out all examples you are not using.

- Set the AIPS user number with variable `user`.[3]
- Set `AIPS_ROOT`, `AIPS_VERSION`, and `DA00` as appropriate for your AIPS installation.
- Set `nThreads` to the number of threads Obit tasks are allowed to spawn. This will improve performance on multi-core machines. `nThreads` should not exceed the number of cores on your machine.
- Specify the AIPS data directory that should be used by setting variable `disk` to a 1-relative index of `adirs`. (Note that `disk` is actually an index to an AIPS array created from `adirs`; AIPS arrays are 1-based.)

A brief explanation of the wrapper command line arguments and options can be found using the `-h` option:

```
$ VLBAContPipeWrap.py -h
```

A complete description of each option is given in Command Line Options, below.

The two required command line arguments are the archive query start and stop dates. As an example, this command will cause the wrapper to query the NRAO archive for VLBA observations from January of 2010:

```
$ VLBAContPipeWrap.py 2010-jan-01 2010-jan-31
```

The wrapper will print a table of the archive response and ask the user to select rows from the table for sequential pipeline processing. Use the `-P` option to limit your search to a specific project code. Use the `-q` option to stop the wrapper after printing the archive response.

After the wrapper downloads a file from the archive it will generate a directory within the current working directory with a name composed of the project code, the 6-digit observation date (YYMMDD), and the archive file ID each separated by underscores. (For example, project BL0149, session AA was observed on 2007-Jun-03 and has archive file ID 235173746; the pipeline directory generated for this file will be `BL0149_070603_235173746`.) Many files will be generated and stored in this directory. These files and their associated metadata are described below in Data Products.

If a destination directory has been specified using environment variable `DESTDIR` or command line option `--destdir` (see Command Line Options) the new data directory will be moved to the destination directory when processing is complete.

[1]When the archive mirror in Charlottesville is complete, users on the NRAO CV Network File System will also be able to use the pipeline. However, some changes to the pipeline's archive interface may be required.

[2]For an Obit binary installation, the path to the scripts directory from the top-level Obit installation directory is `./share/obit/scripts`.

[3]If you intend to run multiple pipeline processes in parallel, it's a good idea to use different AIPS user numbers for each process to ensure there are no conflicts in reading from or writing to the AIPS catalog. Presently, this means setting up a directory with an AIPS setup script for each pipeline process.

4 The Pipeline Wrapper

The continuum pipeline wrapper, `VLBAContPipeWrap.py`, simplifies the job of starting the pipeline by

- providing a simple interface to the NRAO VLBA Archive,
- automatically downloading data to a directory on the NRAO AOC network,
- setting up the input parameters for the pipeline script,
- executing the pipeline script, and
- copying the data to a storage directory when finished.

4.1 Command Line Options

The command line options for the pipeline wrapper are described here.

Option	**Description**
-h, --help	Displays a brief help message that describes command-line arguments and options.
-P PROJECT, --project=PROJECT	Queries the archive for a specific project code.
-q, --query	Performs a query and prints the archive response summary only. Does not setup directories for processing or start pipeline processes.
-a, --all	Automatically processes all files in archive response. Requires no human interaction.
-m, --metadata	Prints the usual summary of the archive response and then prints all metadata as a list of Python dictionaries. (The summary contains only a subset of the response metadata.)
-i, --ignoreidi	Ignores all FITS IDI files in archive response. By default, FITS-IDI files are listed only for new-correlator observations, taken on or after 2009 December 10.
--showallidi	Shows all FITS IDI files in archive response. By default, FITS-IDI files are printed only for new-correlator observations, taken on or after 2009 December 10.

... continued on next page

Option	Description
--multiidi	Downloads and fills multiple old-correlator FITS-IDI files. This option allows for processing old-correlator FITS-IDI files rather than the pipeline-processed FITS-AIPS files. The user should select one or more FITS-AIPS files; the wrapper will then download and fill the corresponding FITS-IDI files automatically. The correspondence is determined by the start and end time of the FITS-AIPS file.
-F, --finish	Assumes data have already been pipeline processed. Skips data download and processing; verifies the pipeline data-file manifest and moves data to the destination directory, if it has been specified.
--destdir=DESTDIR	Moves pipeline data products to DESTDIR when processing has finished and manifest has been verified.

4.2 Loading multiple FITS-IDI files

The VLBA Archive currently contains data from two correlators. The new DiFX correlator outputs single FITS-IDI files for each observing session. The old- correlator output one or more FITS-IDI files for each observing session. These files require special handling for reduction in Obit (or AIPS). For this reason, the raw IDI files were processed (by a different pipeline) to produce a single FITS-AIPS file for each observing session.

The VLBA pipeline is designed to process one data file at a time. For old-correlator data, this means processing FITS-AIPS files; for new-correlator data, this means processing FITS-IDI files. However, some of the old-correlator FITS-AIPS files contain errors that can be avoided by using the original FITS-IDI files directly. The wrapper has therefore been enhanced with a `--multiidi` option to allow for automated retrieval, concatenation, and processing of multiple IDI files.

NOTE: The wrapper currently does not work when old-correlator FITS-IDI files are selected directly. To load data from old-correlator FITS-IDI files, use the `--multiidi` option and *select the corresponding FITS-AIPS files.*

5 The Pipeline Script

The continuum pipeline can be run manually by invoking `VLBAContPipe.py` as an ObitTalk script. This allows you to restart the pipeline at any point, should it crash. It also allows you to rerun a subset of the pipeline by turning on or off various steps in the pipeline process.

To run the pipeline manually, two input parameter files must be provided as arguments on the command line:

```
$ ObitTalk VLBAContPipe.py AIPSSetup.py PipelineParms.py
```

The first argument to `VLBAContPipe.py` is the AIPS setup Python script. This is the same AIPSSetup.py script described above in Quick Start.

The second command line argument to `VLBAContPipe.py` is the pipeline parameters file. A template of the parameters file is distributed with the pipeline source code in VLBAContTemplateParm.py. To run the pipeline script you should make a local copy of the parameters template and replace all the substitution keys with values appropriate for your data set. Each substitution key is explained at the top of the template file along with a data type where it is not obvious from the context. At the bottom of the parameters file are the pipeline control parameters. These parameters allow the user to:

- turn on debug mode which prints the Obit and AIPS task input parameters prior to task execution and leaves Obit task input files in the `/tmp` directory for debugging,
- specify the type of data file to load: UVFITS (also known as FITS-AIPS) or FITS-IDI,
- adjust pipeline input parameters, and
- turn on or off various steps in the pipeline process.

6 Data Products

The pipeline generates metadata and data files that fall into one of two categories: multi-source data and single-source data. A complete table of file data and metadata products is available online. Some of the most useful data products are described below.

HTML Report (ex: `BL0149_BN_2cm.report.html`) A human-readable report on all metadata and file data products generated in HTML.

Pipeline log (ex: `BL0149_BN_2cm.log`) The VLBA pipeline log file. This is the place to go for diagnosing problems and reviewing pipeline performance.

Clean image, total intensity (ex: `BL0149_BN_2cm_0010+405.IClean.fits`) The self-calibrated clean image. The extension `IClean` signifies that this is the total intensity clean image.

Contour plot (ex: `BL0149_BN_2cm_0010+405.cntr.ps`) A contour plot produced from the total intensity clean image. A version of this plot is generated in PostScript and JPEG formats.

Diagnostic visibility plots (ex: `BL0149_BN_2cm_0010+405.amp.jpg`) Diagnostic plots are generated to show:

- amplitude versus uv-distance,
- uv-coverage (u versus v), and
- visibilities in the complex plane (real versus imaginary).

The diagnostic plots are generated in JPEG format.

Calibrated and averaged uv data (ex: `BL0149_BN_2cm.CalAvg.uvtab`) The calibrated and averaged visibility data.

Calibrated AIPS tables (ex: `BL0149_BN_2cm.CalTab.uvtab`) The calibrated AIPS tables without visibility data.

7 Troubleshooting

Problem: One of the AIPS tables contains an error that crashes the pipeline or produces erroneous results.

Solution: If you loaded data from an old-correlator (observed before 2009-Dec-11) pipeline-generated FITS-AIPS file, first try downloading the original FITS-IDI files, and running the pipeline on those files directly. This can be done by using the `--multiidi` option for the pipeline wrapper. In some cases errors that appear in the FITS-AIPS files are not present in the original FITS-IDI files.

If this does not resolve the problem, or if the error is present in FITS-IDI data produced by the DiFX correlator (2009-Dec-11 or later) there is no easy fix. Your best bet in this case is to correct the error manually and run the pipeline on the corrected data. Refer to the appropriate Obit and AIPS documentation for instructions on how to do this. Remember that you can turn various parts of the pipeline on or off by editing the pipeline parameters file described in The Pipeline Script.

A list of VLBA data files that cannot be processed using the VLBA pipeline or that require special handling is available online. If you find a file that you believe should be added to this list please email the authors.

8 Appendices

8.1 Python Modules

The pipeline consists of several Python modules, described here.

VLBACal.py A collection of functions that perform various steps in the reduction process. Typical functions setup and invoke Obit or AIPS tasks to accomplish the data reduction.

VLBAContPipe.py The VLBA continuum pipeline. See section The Pipeline Script for details.

VLBAContPipeWrap.py A wrapper for the continuum pipeline. See section The Pipeline Wrapper for details.

VLBAContTemplateParm.py A template Python file used as input to VLBAContPipe.py. The wrapper inserts appropriate values in this template for each execution of the continuum pipeline.

VLBALinePipe.py A development version of the VLBA spectral line pipeline. This module is not yet functional.

PipeUtil.py A collection of functions that perform various pipeline-related tasks.

IDIFix.py A function that fixes old-correlator (pre-2010) FITS-IDI files.

mjd.py A class that converts dates between Gregorian and Modified Julian formats.

8.2 Archiving Data Products

Authorized NRAO staff may wish to commit data products to the NRAO archive. The pipeline output files have been designed in coordination with the archive operator, John Benson <jbenson@nrao.edu>, to make this easy. Contact John Benson to discuss the creation of a staging directory for automated archive ingestion. Once a staging directory is agreed upon, simply copy your pipeline data directories into the staging directory to have them ingested into the archive.

Pipeline data products should be validated manually before they are put into the archive. For this reason we recommend you not use the `DESTDIR` or `--destdir` functionality to automatically copy data directly to the staging area. Doing so may result in bad data being ingested into the archive.